Lab2報告

組員：108552005 高儀津(Luca\_Kao) , 108552017 申自強(Arthur.Shen)

1. A list of structure how script on Kaggle.

Ans. 1.1 由於本次資料集有58.82 GB 過大光下載在組員間就花了兩天的時間 , 期間一邊下載一邊思考這次的作業有別於上次單純的csv 檔, 還外加圖片檔, 所以資料集才會這麼大 , 在研究資料集圖片後我們這組決定同組的兩個人使用不同的作法 , 因為兩個人擁有的硬體資源不同 :

108552005 高儀津 : 使用本地訓練(Nvdia GPU 1070)和跟實驗室借用 Nvdia GPU 2070 , 採用Pytorch的方式做運算 .

108552017 曾自強 : 主要使用kaggle的資源和本地訓練居多 , 採用TensorFlow的方式做運算 .

曾自強在做 TensorFlow嘗試幾種方式有遇到以下困難和嘗試解決的做法

1. 直接做kaggle 作業, kaggle 提供每週30小時GPU的使用時間和一個kernel 的

section 9 小時的執行限制.

2. 下載資料集至 Google Drive 並且掛在個人硬碟至Google Colab上執行.使用時間僅可連續

12小時 .

3. 使用Kaggle jason API 續傳資料到本地訓練. 使用時間無限, 平日白天還是要上班,就算遠端

作業學習效果還是有限.

我選第三,但因為下載實在太慢, 所以同時間我也做了其他兩項, 只是這次的圖片分類我用了最

基本的CNN網路還是超過kaggle 的九小時使用限制. 而Goolge Colab 在下載圖片時因為

Kaggle jason工具的關係,也無法一直續載.

這部份要再想想有沒有更好的辦法？？

1.2 本地TensorFlow GPU的啟動一直失敗!!主要是用Anaconda 安裝的 jupyter 沒有正確的將

CNN的TensorFlow 函式庫動態載進來.因此也花了不少時間.

2.1 訓練模型的過程一直無法確定方向,可是每次訓練又要很長時間, 自己因為上班總是要中斷訓

練過程. 因此開始思考如何保存訓練中的模型,並在下一次訓練繼續下次.

作法是

1. 設checkpoint, 如下

checkpoint\_path = "training\_1/cp.ckpt"

checkpoint\_dir = os.path.dirname(checkpoint\_path)

cp\_callback =

tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint\_path,

save\_weights\_only=True,

verbose=1)

model.fit(train\_images,

train\_labels,

epochs=10,

validation\_data=(test\_images,test\_labels),

callbacks=[cp\_callback]) # Pass callback to training

2. Model Load wights, 如下

model.load\_weights(checkpoint\_path)

3. Save Model, 如下

model.save('saved\_model/my\_model')

4. Load Model, 如下

new\_model =

tf.keras.models.load\_model('saved\_model/my\_model')

3.1 本地的GPU 記憶體只有8G, 常常一載入TensorFlow 的CNN模組就Error 執行不下去.

作法是

在script 前加上

physical\_devices = tf.config.experimental.list\_physical\_devices('GPU')

assert len(physical\_devices) > 0, "Not enough GPU hardware devices available"

config = tf.config.experimental.set\_memory\_growth(physical\_devices[0], True)

這樣Juypter note就可以順利執行.

2.Key concepts of training

Ans: 2.1 這次的作業是參考Kaggle 上已有的Kernel 外加自己在書本及TensorFlow的範例為思路.一開

始分析資料集為圖片 ＋ Metadata 的 Jason 敘述檔.由於Jason 敘述檔包含每張圖片的大小 、類別、

檔案路徑. 所以使用Panda 的dataframe 將其整理為單一的 frame.

#Combine DataFrames

train\_data = train\_metadata.merge(train\_categories, on='category\_id', how='outer')

train\_data = train\_data.merge(train\_images, on='image\_id', how='outer')

train\_data = train\_data.merge(train\_regions, on='region\_id', how='outer')

2.2 在圖片的預處理上有兩種方法, 分別是fit 與使用fit\_generator.兩者使用上主要差別fit直接

將已存在numpy 的影像資料餵給Model, 這樣做是一次將圖片檔讀進記憶體中, 效率快但記憶體需

求大, 遇上訓練資料大的時候,容易造成記憶體overload. 至於fit\_generator則是一部一部份的將每

次迭代所需要的資料由資料集讀進ImageDataGenerator, 在餵給Model. 除了可以節省記憶體外,

在ImageDataGenerator的過程中還可以對圖片做縮放、強化等特徵調變.

train\_datagen = datagen\_with\_augmentation.flow\_from\_dataframe(dataframe=train\_data,

directory=train\_images\_dir,

x\_col='file\_name',

y\_col='category\_id',

class\_mode="raw",

batch\_size=batch\_size,

color\_mode = 'rgb',

target\_size=(img\_height,img\_width)

)

history = model.fit\_generator(train\_datagen\_wrapper,

epochs=epochs\_num,

validation\_data=val\_datagen\_wrapper,

steps\_per\_epoch=steps\_per\_epoch,

validation\_steps=steps\_per\_epoch)

2.3 模型選用, 考慮自己還是先將學習基礎打穩, 所以就拿書本上CNN及TensorFlow上的範例做研

究.且本地的記憶體有限, 故將每次的迭代所需的batch\_size 由一般的32 設為 16.避免在訓練過程中

失敗.

2.4模型建立, 扣除第一層Input layer, 有三層Conv2D, 三層池化層MaxPooling2D, 二層Dropout

做正規化, 與最後的Output layer 如以下

model = Sequential()

model.add(Conv2D(64, kernel\_size=5, activation='relu',...)

model.add(Conv2D(64, kernel\_size=5, activation='relu',...)

model.add(MaxPooling2D(2, 2))

model.add(Conv2D(128, kernel\_size=3, activation='relu',...)

model.add(MaxPooling2D(2, 2))

model.add(Dropout(0.25))

model.add(Conv2D(128, kernel\_size=3, activation='relu',...)

model.add(MaxPooling2D(2, 2))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(num\_classes / 100))

model.add(Dropout(0.2))

model.add(Dense(num\_classes, activation='softmax')

2.5模型效果, 效能不好正確率只有0.0018% ~ 0.0020%, 看來我的學習方向還要調整.

補充 :

Kaggle上最後採用的成績是以PyTorch訓練 20 epoch 得到0.31585為主 , 但由於不管PyTorch和Tensor-flow 整體上訓練model 的架構都大致相同 , 在此不詳述PyTorch的作法(參考108552005) , 本份報告只寫Tensor-flow部分

|  |
| --- |
| **The end of report all prcocess in charge of as below**  **Data Cleaning: Both**  **Feathure selection: Both**  **Python coding: Both**  **Short report: Both**  **Kaggle Model: zip**檔內  **HW-link :<https://github.com/lars10192002/Introduction_to_Deep_Learning>**  **Kaggle :** <https://www.kaggle.com/c/herbarium-2020-fgvc7/leaderboard> |